

Foundation of Cryptography, Lecture 10

Secure Computation, *Two Parties*¹

Handout Mode

Iftach Haitner

Tel Aviv University.

December 31, 2025

¹ Prepared: 2025/12/31,09:27:08.

Section 1

The Model

Multiparty Computation

- ▶ Multiparty Computation – computing a functionality f
- ▶ Secure Multiparty Computation: compute f in a “secure manner”
 - ▶ Correctness
 - ▶ Privacy
 - ▶ Independence of inputs
 - ▶ Guaranteed output delivery
 - ▶ Fairness : corrupted parties should get their output iff the honest parties do
 - ▶ and ...
- ▶ Examples: coin-tossing, broadcast, electronic voting, electronic auctions
- ▶ How should we model it?
- ▶ Real Vs. Ideal paradigm

Real-model execution

For a protocol $\pi = (A, B)$ and inputs $x_C, x_A, x_B \in \{0, 1\}^*$, let $\text{REAL}_\pi(x_C, x_A, x_B)$ be the **joint** output of $(A(x_A), B(x_B))(x_C)$.

Given a two-party protocol π , an algorithm taking the role of one of the parties in π is:

- ▶ Honest. Acts **exactly** according to π .
- ▶ Semi-honest. Acts honestly but (1) Might prematurely abort (2) Outputs its *view*.
- ▶ Malicious. Acts **arbitrarily**.

π' is **admissible** with respect to π , if at least one party is honest.

Ideal model execution

For a pair of oracle-aided algorithms $\hat{\pi} = (\hat{A}, \hat{B})$, inputs $x_c, x_{\hat{A}}, x_{\hat{B}} \in \{0, 1\}^*$ and a function $f = (f_{\hat{A}}, f_{\hat{B}})$, let $\text{IDEAL}_{\hat{\pi}}^f(x_c, x_{\hat{A}}, x_{\hat{B}})$ be the joint output of the parties in the end of the following experiment:

1. The input of \hat{P} , $\hat{P} \in \{\hat{A}, \hat{B}\}$, is $(x_c, x_{\hat{P}})$.
2. \hat{P} sends $y_{\hat{P}}$ to the **trusted party**.
3. Trusted party sends $z_{\hat{P}} = f_{\hat{P}}(y_{\hat{A}}, y_{\hat{B}})$ to \hat{P} in an arbitrary order.
 - after receiving its output, a party can instruct trusted party to **abort**: not send the output to other party.
4. Each party outputs some value.

An oracle-aided algorithm is:

- ▶ Honest. Sends its private input to the trusted party (i.e., sets $y_{\hat{P}} = x_{\hat{P}}$), and its only output is the value it gets from the trusted party (i.e., $z_{\hat{P}}$).
- ▶ Semi-honest. Acts honestly but (1) Might ask the trusted party to abort (2) Outputs its view.
- ▶ Malicious. Acts **arbitrarily**.

$\hat{\pi}$ is **admissible**, if at least one party is honest.

Secure computation

Definition 1 (Secure computation)

A PPT protocol $\pi = (A, B)$ **securely computes** f , if \forall admissible PPT protocol $\pi' = (A', B')$, exists admissible PPT pair $\hat{\pi} = (\hat{A}, \hat{B})$, s.t.

$$\{\text{REAL}_{\pi'}(x_C, x_A, x_B)\}_{x_C, x_A, x_B \in \{0,1\}^*} \approx_c \{\text{IDEAL}_{\hat{\pi}}^f(x_C, x_A, x_B)\}_{x_C, x_A, x_B \in \{0,1\}^*}$$

In case π' is honest, we require that $\hat{\pi}$ is honest, and the ensembles to be identical.

- ▶ Recall that the enumeration index (i.e., x_C, x_A, x_B) is given to the distinguisher.
- ▶ π securely computes f implies that π computes f **correctly**.
- ▶ Security parameter
- ▶ Auxiliary inputs
- ▶ We start with presenting a semi-honest secure protocol for **any** two-party functionality, and then **compile** it into a maliciously secure protocol.

Section 2

Coin flipping

Coin flipping

An coin-flipping protocol **securely computes** the functionality

$\text{CF}(1^\ell) = (U_\ell, U_\ell)$ for $U_\ell \xleftarrow{\mathcal{R}} \{0, 1\}^\ell$.

- We will focus on single bit.

The protocol

- ▶ Let Com be a perfectly binding, NI, cmt.
- ▶ Let (P^1, V^1) be a ZKP-POK for $\mathcal{L}^1 = \{((1^n, c), (b, r)) : c = \text{Com}(1^n, b; r)\}$.
- ▶ Let (P^2, V^2) be a ZKP for $\mathcal{L}^2 = \{((1^n, c, b), r) : c = \text{Com}(1^n, b; r)\}$.

Protocol 2 (Coin flipping (A, B))

Common input: 1^n .

1. A Sample $b_A \xleftarrow{R} \{0, 1\}$ and $r \xleftarrow{R} \{0, 1\}^n$, and send $c \leftarrow \text{Com}(b_A; r)$.
2. The parties interact in $(P^1(b_A, r), V^1)(1^n, c)$.
3. B: Send $b_B \xleftarrow{R} \{0, 1\}$.
4. A: Send b_A .
5. The parties interact in $(P^2(r), V^2)(1^n, c, b_A)$.
6. Each party outputs $b \leftarrow b_A \oplus b_B$.

Claim 3

Protocol 2 securely computes CF (with abort).

Proving Claim 3

We need to prove that \forall (admissible) PPT protocol $\pi' = (A', B')$ for (A, B) , exists oracle-aided PPT pair $\hat{\pi} = (\hat{A}, \hat{B})$ s.t.

$$\{\text{REAL}_{\pi'}(1^n)\} \approx_c \{\text{IDEAL}_{\hat{\pi}}^{\text{CF}}(1^n)\}, \quad (1)$$

where the enumeration is over $n \in \mathbb{N}$.

Malicious A'

Define the malicious oracle-aided strategy \hat{A} as follows:

Algorithm 4 (\hat{A})

input: 1^n .

1. Call CF, let b be the output.
2. Emulate $(A', B)(1^n)$ for the first two steps, and extract b_A from A' .
3. Continue the emulation till its end, while changing b_B to $b \oplus b_A$.
4. Abort CF if the executions aborts.

Let $\pi' = (A', B)$ and $\hat{\pi} = (\hat{A}, \hat{B})$, where \hat{B} is honest.

Claim 5

$$\{\text{REAL}_{\pi'}(1^n)\} \approx_c \{\text{IDEAL}_{\hat{\pi}}^{\text{CF}}(1^n)\}.$$

Proof?

Malicious B'

Let S_2 be the (BB) simulator for (\cdot, V_2) . Define the malicious oracle-aided strategy \hat{B} as follows:

Algorithm 6 (\hat{B})

input: 1^n .

1. Call CF, let b be the output.
2. Emulate $(A, B')(1^n)$ for the first four steps.
3. Send $b_A \leftarrow b \oplus b_B$ as B 's answer.
4. Continue the emulation till its end, while changing b_A to $b \oplus b_B$, and simulating B' 's view in the execution of (P^2, V_2) , where it acts as the verifier, using $S_2^{B'}(1^n, c, b_A)$.
5. Abort CF if the executions aborts.

Let $\pi' = (A, B')$ and $\hat{\pi} = (\hat{A}, \hat{B})$, where \hat{A} is honest.

Claim 7

$$\{\text{REAL}_{\pi'}(1^n)\} \approx_c \{\text{IDEAL}_{\hat{\pi}}^{\text{CF}}(1^n)\}.$$

Proof? where did we use the zero-knowledge of (P_1, V_{C1}) ?

Section 3

Oblivious Transfer

Oblivious transfer

An (one-out-of-two) OT protocol **securely computes** the functionality $\text{OT} = (\text{OT}_S, \text{OT}_R)$ over $(\{0, 1\}^* \times \{0, 1\}^*) \times \{0, 1\}$, where $\text{OT}_S(\cdot) = \perp$ and $\text{OT}_R((\sigma_0, \sigma_1), i) = \sigma_i$.

- ▶ “Complete” for multiparty computation
- ▶ We show how to construct for bit inputs.

Oblivious transfer from trapdoor permutations

Let (G, f, Inv) be a TDP and let b be an hardcore predicate for f .

Protocol 8 $((S, R))$

Common input: 1^n

S's input: $\sigma_0, \sigma_1 \in \{0, 1\}$.

R's input: $i \in \{0, 1\}$.

1. S chooses $(e, d) \leftarrow G(1^n)$, and sends e to R .
2. R chooses $x_0, x_1 \leftarrow \{0, 1\}^n$, sets $y_i = f_e(x_i)$ and $y_{1-i} = x_{1-i}$, and sends y_0, y_1 to S .
3. S sets $c_j = b(\text{Inv}_d(y_j)) \oplus \sigma_j$, for $j \in \{0, 1\}$, and sends (c_0, c_1) to R .
4. R outputs $c_i \oplus b(x_i)$.

Claim 9

Protocol 8 securely computes **OT** (in the semi-honest model).

Proving Claim 9

We need to prove that \forall semi-honest (admissible) PPT protocol $\pi' = (S', R')$ for (S, R) , exists oracle-aided PPT pair $\hat{\pi} = (\hat{S}, \hat{R})$ s.t.

$$\{\text{REAL}_{\pi'}(1^n, (\sigma_0, \sigma_1), i)\} \approx_c \{\text{IDEAL}_{\hat{\pi}}^{\text{OT}}(1^n, (\sigma_0, \sigma_1), i)\}, \quad (2)$$

where the enumeration is over $n \in \mathbb{N}$ and $\sigma_0, \sigma_1, i \in \{0, 1\}$.

R's security

For a semi-honest S' , define semi-honest oracle-aided strategy \hat{S} as follows:

Algorithm 10 (\hat{S})

input: $1^n, \sigma_0, \sigma_1$

1. Send (σ_0, σ_1) to the trusted party.
2. Emulate $(S'(1^n, \sigma_0, \sigma_1), R(1^n, 0))$.
3. Output the output that S' does.

Let $\pi' = (S', R)$ and $\hat{\pi} = (\hat{S}, \hat{R})$, where \hat{R} is honest.

Claim 11

$$\{\text{REAL}_{\pi'}(1^n, (\sigma_0, \sigma_1), i)\} \equiv \{\text{IDEAL}_{\hat{\pi}}^{\text{OT}}(1^n, (\sigma_0, \sigma_1), i)\}.$$

Proof?

S's security

For a semi-honest implementation R' of R , define the oracle-aided semi-honest strategy \hat{R} as follows.

Algorithm 12 (\hat{R})

input: $1^n, i \in \{0, 1\}$,

1. Send i to the trusted party, and let σ be its answer.
2. Emulate $(S(1^n, \sigma_0, \sigma_1), R'(1^n, i))$, for $\sigma_i = \sigma$ and $\sigma_{1-i} = 0$.
3. Output the output that R' does.

Let $\pi' = (S, R')$ and $\hat{\pi} = (\hat{S}, \hat{R})$, where \hat{S} is honest.

Claim 13

$$\{\text{REAL}_{\pi'}(1^n, (\sigma_0, \sigma_1), i)\} \approx_c \{\text{IDEAL}_{\hat{\pi}}^{\text{OT}}(1^n, (\sigma_0, \sigma_1), i)\}.$$

Proof?

Section 4

Yao Garbled Circuit

Before we start

- ▶ Fix a (multiple message) semantically-secure private-key encryption scheme (G, E, D) with
 1. $G(1^n) = U_n$.
 2. For any $m \in \{0, 1\}^*$
$$\Pr_{(d, d') \leftarrow (\{0, 1\}^n)^2} [D_d(E_{d'}(m)) \neq \perp] = \text{neg}(n).$$
- ▶ Can we construct such a scheme?

Yes, append 0^n at the end of the message.
Or use the MAC-based CCA2 scheme
- ▶ Boolean circuits: gates, wires, inputs, outputs, values, computation

The Garbled Circuit

Fix a Boolean circuit C and $n \in \mathbb{N}$.

- ▶ Let \mathcal{W} and \mathcal{G} be the (indices) of **wires** and **gates** of C , respectively.
- ▶ For $w \in \mathcal{W}$, associate a pair of random ‘keys’ $k_w = (k_w^0, k_w^1) \in (\{0, 1\}^n)^2$.
- ▶ For $g \in \mathcal{G}$ with input wires i and j , and output wire h , let $T(g)$ be the following table:

input wire i	input wire j	output wire h	hidden output wire
k_i^0	k_j^0	$k_h^{g(0,0)}$	$E_{k_i^0}(E_{k_j^0}(k_h^{g(0,0)}))$
k_i^0	k_j^1	$k_h^{g(0,1)}$	$E_{k_i^0}(E_{k_j^1}(k_h^{g(0,1)}))$
k_i^1	k_j^0	$k_h^{g(1,0)}$	$E_{k_i^1}(E_{k_j^0}(k_h^{g(1,0)}))$
k_i^1	k_j^1	$k_h^{g(1,1)}$	$E_{k_i^1}(E_{k_j^1}(k_h^{g(1,1)}))$

Figure: Table for gate g , with input wires i and j , and output wire h .

The Garbled Circuit, cont.

input wire i	input wire j	output wire h	hidden output wire
k_i^0	k_j^0	$k_h^{g(0,0)}$	$E_{k_i^0}(E_{k_j^0}(k_h^{g(0,0)}))$
k_i^0	k_j^1	$k_h^{g(0,1)}$	$E_{k_i^0}(E_{k_j^1}(k_h^{g(0,1)}))$
k_i^1	k_j^0	$k_h^{g(1,0)}$	$E_{k_i^1}(E_{k_j^0}(k_h^{g(1,0)}))$
k_i^1	k_j^1	$k_h^{g(1,1)}$	$E_{k_i^1}(E_{k_j^1}(k_h^{g(1,1)}))$

Let \mathcal{I} and \mathcal{O} be the input and outputs wires of \mathcal{C} .

- ▶ For $g \in \mathcal{G}$, let $\tilde{T}(g)$ be a **random permutation** of the fourth column of $T(g)$.
- ▶ For $w \in \mathcal{W}$, let $\mathbf{C}(x)_w$ be the **bit-value** computation of $\mathbf{C}(x)$ assigns to w
- ▶ Given
 1. $\tilde{T} = \{(g, \tilde{T}(g))\}_{g \in \mathcal{G}}$.
 2. $\{(w, k_w^{\mathbf{C}(x)_w})\}_{w \in \mathcal{I}}$ for some x .
 3. $\{(w, k_w = (k_w^0, k_w^1))\}_{w \in \mathcal{O}}$.

One can efficiently compute $\mathbf{C}(x)$.

- ▶ (essentially) The above leaks no additional information about x !

Example, GV for OR

On board...

The protocol

- ▶ Let $f(x_A, x_B) = (f_A(x_A, x_B), f_B(x_A, x_B))$ be a function, and let C be a circuit that computes f .
- ▶ Let \mathcal{I}_A and \mathcal{I}_B be the input wires corresponds to x_A and x_B respectively in C , and let \mathcal{O}_A and \mathcal{O}_B be the output wires corresponds to f_A and f_B outputs respectively in C .
- ▶ Recall that $C(x)_w$ is the bit-value the computation of $C(x)$ assigns to w .
- ▶ Let (S, R) be a secure protocol for OT.

Protocol 14 ((A, B))

Common input: 1^n . **A/B's input:** x_A/x_B

1. **A:** Sample at random $\{k_w = (k_w^0, k_w^1)\}_{w \in \mathcal{W}}$, and generate \tilde{T} .
2. **A:** Send \tilde{T} , $\{(w, k_w^{C(x_A, \cdot)_w})\}_{w \in \mathcal{I}_A}$, and $\{(w, k_w)\}_{w \in \mathcal{O}_B}$ to **B**.
3. $\forall w \in \mathcal{I}_B$: The parties interact in $(S(k_w), R(C(\cdot, x_B)_w))(1^n)$.
4. **B:** Compute the (garbled) circuit, and send $\{(w, k_w^{C(x_A, x_B)_w})\}_{w \in \mathcal{O}_A}$ to **A**.
5. The parties compute $f_A(x_A, x_B)$ and $f_B(x_A, x_B)$ respectively.

Example, protocol for OR

On board...

Claim 15

Protocol 14 securely computes f (in the semi-honest model)

Proof: We focus on the security of A . For a semi-honest B' , define

Algorithm 16 (\hat{B})

input: 1^n and x_B .

1. Send x_B to the trusted party, and let o_B be its answer.
2. Emulate the first 4 steps of $(A(0^{|x_A|}), B'(x_B)(1^n))$.
3. For each $w \in \mathcal{O}_B$: permute the order of the pair k_w according to o_B , and the key of w computed in the emulation.
4. Complete the emulation, and output the output that B' does.

Claim: \hat{B} is a good “simulator” for B' .

Security of B ?

Extensions

- ▶ Efficiently computable f

Both parties first compute C_f – a circuit that compute f for inputs of the right length

- ▶ Hiding C ? All but its size

Section 5

Malicious Security

Semi-honest to malicious security

Let $\pi^h = (A^h, B^h)$ semi-honest protocol for f . The parties prove that they act “according to π^h ”:

1. Forces the parties to chose their random coin properly
2. Before each step, the parties prove in \mathcal{ZK} that they followed the prescribed protocol (with respect to the random-coins chosen above)

We will use

1. Perfectly binding POK. NI commitment Com . (?)
2. Zero-knowledge proof for (soon to be implicitly defined) $\mathcal{L} \in \mathcal{NP}$ with negligible soundness and completeness error
3. Let $\ell(n)$ bound the number of random coins the parties of π^h use on common input of length n .

The compiler

Protocol 17 (Maliciously secure protocol $\pi = (A, B)$)

Common input: x_c , let $n = |x_c|$

P's input: x_P .

1. For each $\hat{P} \in \{A, B\}$:
 - 1.1 \hat{P} commits to x_P .
 - 1.2 \hat{P} commits to $r_P^1 \leftarrow \{0, 1\}^\ell$.
 - 1.3 Other party sends $r_P^2 \leftarrow \{0, 1\}^\ell$ to \hat{P}
 - 1.4 \hat{P} sets $r_P = r_P^1 \oplus r_P^2$.
2. The parties interact $(A^h(x_A; r_A), B^h(x_B; r_B))(x_c)$:
 - 2.1 After party \tilde{P} sends a message m , it **proves** in ZK that m is what P^h would send on input x_P , randomness r_P , and previous messages received.
 - 2.2 The other party abort, if proof fails.

Claim 18

Assume π^h is semi-honest secure for f , then π is **maliciously** secure for f (w/ abort).